July 7, 2010

Numerical Method in Classical Dynamics

Charles Schwartz[*]

Department of Physics, University of California
Berkeley, California 94720

Abstract

A set of algorithms is presented for efficient numerical calculation of the time evolution of classical dynamical systems. Starting with a first approximation for solving the differential equations that has a "reversible" character, we show how to bootstrap easily to higher order accuracy.

---

[*]E-mail: schwartz@physics.berkeley.edu

1 The Problem

We start by considering Newton's Law of motion in one dimension, written as a pair of first order time evolution equations.

$$\frac{d}{dt}\left(\begin{array}{c} x(t) \\ v(t) \end{array}\right) = \left(\begin{array}{c} v(t) \\ f(x(t)) \end{array}\right) = M\left(\begin{array}{c} x(t) \\ v(t) \end{array}\right) \tag{1.1}$$

where both x and v are time dependent functions to be determined at some later time t, given their values at some initial time $t = 0$. The force is given by some specified function $f(x)$; and the quantity M is defined as the (non-linear) matrix/operator specified above.

For simplicity I will write

$$\psi(t) = \left(\begin{array}{c} x(t) \\ v(t) \end{array}\right). \tag{1.2}$$

We assume that there exists an operator $E(tM)$ that is the exact propagator:

$$\psi(t) = E(tM)\,\psi(0), \qquad E(tM) = \lim_{N\to\infty}(1 + \frac{t}{N}M)^N. \tag{1.3}$$

The addition property, $E(t_1 M)E(t_2 M) = E((t_1 + t_2)M)$ follows. Alternatively, we may write,

$$\frac{d}{dt}E(tM) = M\,E(tM). \tag{1.4}$$

This formalism is familiar in the case where M is a general linear operator, and E is simply the ordinary exponential function; however, it is also appropriate for non-linear operators, as derived in reference [1].

Our objective is to show simple and accurate approximations to the operator $E(\delta M)$, for small time-steps δ, for use in automated computations.

2 The General Method

Following the general method given in [1], we start by constructing an approximate propagation operator $R(\delta)$ with the following properties:

$$R(\delta)\,R(-\delta) = 1, \tag{2.1}$$

and

$$R(\delta) = E(\delta M + \delta^3 X_3 + \delta^5 X_5 + \ldots). \tag{2.2}$$

Rather than expanding the approximate result $\psi(t + \delta) \approx R(\delta) \ \psi(t)$ in a power series in δ, we represent R as the exact propagator for some other problem, which is expanded about the true one: based upon M. The restriction (2.1) means that only odd powers of δ occur in the expansion (2.2). The quantities X_3, X_5, etc., are unknown. Our method will show how to eliminate those higher order errors, step by step.

One more general property of the abstract propagator function E is the following.

$$E(A)E(B) = E(A + B + \frac{1}{2}[A/B] + \frac{1}{12}[(A - B)/[A/B]]\ldots), \qquad (2.3)$$

$$E(A)E(B)E(A) = E(2A + B - \frac{1}{6}[(A + B)/[A/B]]\ldots). \qquad (2.4)$$

This is the nonlinear extension of the Baker-Campbell-Hausdorff theorem for the product of exponentials of non-commuting linear operators. The only difference is that, instead of the commutator $[A, B] = AB - BA$ for linear operators, we have the "slash commutator" $[A/B] = A/B - B/A$ for nonlinear operators, as defined in reference [1].

Now we proceed. The initial operator $R(\delta)$ is correct to order δ^2 and so we call it $R_2(\delta)$. Now we construct the following sandwich:

$$R_4(\delta) = R_2(\beta\delta) \ R_2(\gamma\delta) \ R_2(\beta\delta); \qquad (2.5)$$

and try to choose the constants β, γ so that

$$R_4(\delta) = E(\delta M + \delta^5 Y_5 + \ldots). \qquad (2.6)$$

By working with Equation (2.4) we find the simple requirements,

$$2\beta + \gamma = 1, \quad 2\beta^3 + \gamma^3 = 0, \quad \beta = (2 - 2^{1/3})^{-1}, \quad \gamma = -2^{1/3} \ \beta. \quad (2.7)$$

This new formula (2.5) may be read as follows: Take a step forward of length $1.351207\ldots\delta$, then take a step backwards of length $1.702414\ldots\delta$, then another step forward of length $1.351207\ldots\delta$. The result will be one step forward of length δ - with errors of order δ^5.

3 Choosing R

The real challenge now is to construct $R(\delta)$, seemingly accurate only to first order in δ but restricted by the requirement (2.1).

Here is one suggestion, for the particular problem we started with (1.1), that is built in the "sandwich" manner.

$$R_2(\delta) = D_x(\delta/2)D_v(\delta)D_x(\delta/2) \tag{3.1}$$

$$D_x(\delta)\begin{pmatrix} x \\ v \end{pmatrix} = \begin{pmatrix} x + \delta v \\ v \end{pmatrix} \tag{3.2}$$

$$D_v(\delta)\begin{pmatrix} x \\ v \end{pmatrix} = \begin{pmatrix} x \\ v + \delta f(x) \end{pmatrix}. \tag{3.3}$$

It should be apparent that this formulation is very easy to program for automated computation. On the other hand, it is rather cumbersome if one writes out explicit formulas for the overall result of this sequence of operations.

4   Numerical Examples

I have applied this method to a simple problem, the Kepler orbit in a plane. With the initial conditions $x(0) = 1, y(0) = 0, v_x(0) = 0, v_y(0) = 1$, I broke a complete orbit into N steps and saw what was the resulting error in $y(N)$, which ought to return to zero. The results are shown in the tables below, for various values of N and various values of the source strength g (g=1 gives a circular orbit).

TABLES of computational errors

Using R_2	g=0.625	g=1.0	g=2.5
N=100	2×10^{-1}	8×10^{-3}	2×10^{-2}
N=1,000	2×10^{-3}	8×10^{-5}	3×10^{-4}
N=10,000	2×10^{-5}	8×10^{-7}	3×10^{-6}

Using R_4	g=0.625	g=1.0	g=2.5
N=100	3×10^{-2}	8×10^{-5}	2×10^{-3}
N=1,000	3×10^{-6}	8×10^{-9}	2×10^{-7}
N=10,000	3×10^{-10}	8×10^{-13}	2×10^{-11}

Each increase in the number of steps by a factor of 10 improves the accuracy by a factor of 10^2 if we use R_2 and by a factor of 10^4 if we use R_4.

3

Of course, R_4 requires three times as many operations per step, compared to R_2; but that seems a worthwhile investment since we can use many fewer steps for a given overall accuracy.

One can readily extend this calculational method to involve any number of particles, with any forces.

For comparison, I ran this same calculation using the popular Runge-Kutta method, at second order, and compared the results with those shown above for R_2. Overall, one sees the same rate of improvement in accuracy as N is increased; and this is to be expected. For this particular problem I found that my method gave somewhat better accuracy at each level; but I would not offer that as a general rule without much further study; and I encourage others to try both methods on their own favorite problems. I will say, however, that the programming for my method was considerably simpler than that for the R-K method; and I expect that this aspect of the comparison is even more marked as one goes to the fourth order methods.

What about the Richardson technique? As a general rule, if you calculate something with a small parameter δ and know how it converges to the true answer as $\delta \to 0$, then you can accelerate convergence. For example, if you know

$$A(\delta) = A + \delta^2 X_2 + \delta^4 X_4 + \ldots, \tag{4.1}$$

then you can do two calculations and combine the results as follows.

$$\frac{4}{3}A(\delta/2) - \frac{1}{3}A(\delta) = A + \delta^4 Y_4 + \ldots. \tag{4.2}$$

I used this method on the Kepler calculation, using R_2, and found results that were slightly better than those obtained from using R_4. This appears to be a nice alternative method.

Next I added the Richardson extrapolation to the Runge-Kutta (second order) calculation of the same Kepler problem; and found that the results were far inferior to those just mentioned. There was some improvement in accuracy but significantly less than expected. The reason is that the asymptotic formula (4.1) is incorrect for the Runge-Kutta method: there is a term of order δ^3 that belongs there.

The main lesson from these experiments appears to be that the condition (2.1) on the lowest order approximation, which we might call "reversibility", is important.

4

5   Velocity-dependent Force

Here we start by considering a simple type of velocity-dependent force:

$$\frac{d^2x}{dt^2} = f(x, v) = g(x) + vh(x). \tag{5.1}$$

Now, when we construct the approximate propagator $R_2(\delta)$, the D_x part will be the same as before but the D_v part needs to be changed so as to guarantee that $R(-\delta)R(\delta) = 1$.

The way we do this is to find the exact solution of the simple equation

$$\dot{v} = g + vh, \tag{5.2}$$

where we treat $g = g(x)$ and $h = h(x)$ as constants. The solution is easy:

$$v(t) = v(0) + (v(0) + \frac{g}{h})(e^{ht} - 1) \tag{5.3}$$

and this shows us how to construct the operator D_v.

With this special result we can now address the case of a general $f(x, v)$. We do this by taking another time-derivative of the original equation (5.1).

$$\dot{x} = v, \quad \dot{v} = w, \quad \dot{w} = g(x, v) + wh(x, v) \tag{5.4}$$

$$g(x, v) = v\frac{\partial f(x, v)}{\partial x}, \quad h(x, v) = \frac{\partial f(x, v)}{\partial v}. \tag{5.5}$$

Now we construct the following.

$$R_2(\delta) = D_x(\delta/2)D_v(\delta/2)D_w(\delta)D_v(\delta/2)D_x(\delta/2) \tag{5.6}$$

$$D_x(\delta)\begin{pmatrix} x \\ v \\ w \end{pmatrix} = \begin{pmatrix} x + \delta v \\ v \\ w \end{pmatrix} \tag{5.7}$$

$$D_v(\delta)\begin{pmatrix} x \\ v \\ w \end{pmatrix} = \begin{pmatrix} x \\ v + \delta w \\ w \end{pmatrix} \tag{5.8}$$

$$D_w(\delta)\begin{pmatrix} x \\ v \\ w \end{pmatrix} = \begin{pmatrix} x \\ v \\ w + (w + \frac{g}{h})(e^{h\delta} - 1) \end{pmatrix}. \tag{5.9}$$

This technique also shows us how to handle the simplest first order equation $\dot{x} = f(x)$ by turning it into a second order equation, $\dot{x} = v$ and

5

$\dot{v} = vf'(x)$, and then constructing R_2 as in (3.1). As another option, instead of using

$$D_v(\delta) : v \to ve^{\delta f'(x)} \tag{5.10}$$

one could use

$$D_v(\delta) : v \to v\frac{1 + (\delta/2)f'(x)}{1 - (\delta/2)f'(x)}. \tag{5.11}$$

6   Time-dependent Force

Let's return to the original problem (1.1) and allow the force to be explicitly time dependent.

$$\frac{d}{dt}\begin{pmatrix} t \\ x \\ v \end{pmatrix} = \begin{pmatrix} 1 \\ v \\ f(t,x) \end{pmatrix} = M(t)\begin{pmatrix} t \\ x \\ v \end{pmatrix}. \tag{6.1}$$

The natural guess is for the second order approximate propagator $\tilde{R}_2$ to be built from the original R_2 as follows.

$$\tilde{R}_2(\delta) = D_t(\delta/2)R_2(\delta)D_t(\delta/2), \qquad D_t(\delta) : t \to t + \delta. \tag{6.2}$$

This means that we proceed as before but evaluate the force $f(t, x)$ with the variable t at the midpoint of each time interval. This formulation preserves the property (2.1).

Then we can proceed to construct $\tilde{R}_4$ just as before, using three of these operators $\tilde{R}_2$ with the same weight factors β, γ as in (2.5).

7   Discussion

The method described here appears to be a powerful, simple and versatile tool.

It should be apparent how one can continue to improve the method, going from R_4 to R_6 , etc. While it is not easy to guess in advance what level of accuracy will be most efficient in any given problem, the programming procedures outlined above make it relatively easy to experiment and find the best approach.

It is interesting that the equations (2.7) have another solution, one that goes into the complex plane, as follows.

$$\beta \approx 0.324 \pm 0.135i, \qquad \gamma \approx 0.352 \mp 0.270i. \tag{7.1}$$

Which is the best to use? My guess is that for conservative systems, the real solutions are best but for dissipative systems this complex scheme may be better. This whole area needs further study and experimentation.

References

[1] C. Schwartz, *J. Math. Phys.* **38**, 484 (1997). In that paper I report using the method successfully for numerical calculation of the time-dependent nonlinear Schrodinger equation.